Main features

- User login (Employee or Manager)
- Create vacation requests
- View own requests (Employee)
- View and manage all requests (Manager)
- Approve/Reject requests
- Business validations:
    - Approved vacations that overlap are not allowed
    - Requests for more than 15 consecutive days are automatically rejected
- Responsive interface with Tailwind CSS

General architecture

/backend → ASP.NET Core Web API

/frontend → React + TypeScript (Vite)

Frontend and backend are decoupled and communicate via HTTP (Axios).

Backend (.NET)

Technologies used

- .NET 8 – ASP.NET Core Web API
- Entity Framework Core
- SQLite (local database)
- Swagger (API documentation)

Backend structure

backend/

├── Controllers/

├── Data/

├── Models/

├── DTOs/

├── Services/

├── Program.cs

└── appsettings.json

How to run the backend

Requirements

- .NET SDK 8+

Steps

cd backend/LeaveManagement

dotnet restore

dotnet ef database update

dotnet run

The backend runs by default at:

https://localhost:5001

Seed users (preloaded)

Role	Email	ID
Employee	juan01@gmail.com	1
Manager	rodri02@gmail.com	2
Employee	beydi03@gmail.com	3

Authentication (design decision)

JWT and ASP.NET Identity are not used.

Instead:

- Simple login by email
- The backend returns the user data
- The frontend sends the X-User-Id header in each request

Business rules implemented

- An employee **cannot have approved vacations that overlap**
- Requests for **more than 15 consecutive days** are automatically rejected
- Only managers can approve or reject
- An employee can only delete **pending and their own** requests

These rules live in the **service layer**, not in the controllers.

Frontend (React)

Technologies used

- React 18
- TypeScript
- Vite
- Axios
- Tailwind CSS
- React Context API (auth)

Frontend structure

frontend/

├── src/

│ ├── api/

│ ├── context/

│ ├── pages/

│ ├── types/

│ ├── App.tsx

│ └── main.tsx

└── index.css

How to run the frontend

Requirements

- Node.js 18+

Steps

cd frontend

npm install

npm run dev

The application will be available at:

http://localhost:5173

Communication with the backend

- Centralized Axios (src/api/axios.ts)
- The X-User-Id header is automatically added after login
- Error handling displayed directly in the UI

Frontend role management

- **Employee**
    - Create requests
    - View only your requests
- **Manager**
    - View all requests
    - Approve / Reject

The UI adapts dynamically according to the user's role.

UI/UX

- Tailwind CSS for responsive design
- Simple, clear, and accessible components
- Visual feedback for statuses and errors

Testing

- At least one frontend test is included (form validation)
- The structure allows for easy scaling to more tests

Seed users (preloaded)

Role	Email	ID
Employee	juan01@gmail.com	1
Manager	rodri02@gmail.com	2
Employee	beydi03@gmail.com	3